INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Using Parallel Computing to Reduce CPU Power

Jérôme Galtier

No 3621

Février 1999

THÈME 1

ISSN 0249-6399

Using Parallel Computing to Reduce CPU Power

Jérôme Galtier*

Thème 1 — Réseaux et systèmes
Projet SLOOP

Rapport de recherche n ° 3621 — Février 1999 — 18 pages

Abstract: In this paper, we analyze parallelism as an energy-saving technique for mobile terminal or on-board systems. We show that the expected gains are considerable. However, some natural constraints can appear when applying massively this idea. Therefore, we analyze this concept on a simple model in an environment where either the space for circuitry is restricted, or the total weight of the system, including energy sources, has to be minimized. It turns to give new types of CAD problems. Finally, we analyze fault tolerance issues and solutions in a satellite-like environment.

Key-words: Parallelism; on-board computation; CPU energy consumption.

(Résumé : tsvp)

* CNET-INRIA

Unit´e de recherche INRIA Sophia-Antipolis
2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex (France)
T´el´ephone : 04 93 65 77 77 - International : +33 4 93 65 77 77
T´el´ecopie : 04 93 65 77 65 - International : +33 4 93 65 77 65

Utilisation de l'algorithmique parallèle pour la réduction de l'énergie consommée par les calculs

Résumé : Dans cet article, nous étudions le parallélisme en tant que méthode d'économie d'énergie pour les terminaux mobiles ou les systèmes embarqués. Nous montrons que les gains attendus sont considérables. Cependant, des contraintes naturelles apparaissent lors de la mise en œuvre de cette idée à grande échelle. Nous proposons un modèle simple de prise en compte de ces limitations, en raisonnant soit en espace (i.e. taille totale des circuits intégrés) limité, soit en poids limité, en incluant alors le poids des sources d'énergie. Ces modèles se traduisent naturellement en de nouveaux problèmes d'optimisation. Nous considérons aussi les questions de tolérance aux pannes et leurs réponses dans en environnement de type satellite.

Mots-clé : Parallélisme; calcul embarqué; consommation d'énergie CPU.

In the recent years, a particular emphasis has been put on mobile terminals and associated problems. One of the main issues concerns energy problems. Energy impacts directly both on the weight and the autonomy of the systems, due to the limited power of batteries. Recently, an enhanced interest for energy savings has appeared for small satellite technology, due to the uprising of satellite constellations and other new types of mobile networks.

In this paper, we show how to use ideas issued from parallel processing to reduce the power consumption of a given task. In the past, some techniques have been developed to reduce on-board power.

Spin-down techniques [12] detect periods of inactivity of the system to cut the power. These methods have to detect long-enough idle periods, so that the energy cost of resuming is lower than that of keeping the system active. This method has now been largely implemented but contains some inherent limitations, as explained in the following.

Adapting the speed and voltage to the load of the system [8] is an alternative known technique. When the load is low, the CPU clock frequency decreases along with the voltage, saving thereby energy. This method has been introduced at the operating system level of the application. This allows to address the balance between load and energy consumption on the fly, and is quite practical when the nature of the computation is unpredictable. Nevertheless, using a processor with a large panel of clock speeds make some designers quite insecure, especially because the processor cannot have been tested and certified at any clock speed, and also the transition between two speed/voltage modes leaves some possibility of electrical instability. Further research is therefore necessary before applying this technique in real systems.

However, a large part of modern mobile terminals have to handle a constant load. For instance, a mobile phone has to check permanently the channel for a possible call when in passive mode. Similarly voice encryption, coding, compression and other mechanisms require a constant heavy load. Assuming this type of applictions, our idea manages to reduce the power consumed for a given task (i.e. at constant load). This operation is done at the expense of:

- some additional circuitry,
- the design of parallel algorithms.

We stress here that the weight of the circuitry in modern devices is far less than the weight of batteries. Yet some studies have shown that parallelism can bring very promising power reduction in ASICs [16]. Also, many industrials are concerned with consuming energy in a predicted and controlled fashion when the resources are limited. In the following we give more insight on how to estimate the savings. It turns to give new types of optimization problems. As far as we know, they have not been investigated before.

This paper is organized as follows. In Section 1, we show how parallelism may impact energy savings, and how various degrees of parallelism may be handled. In Section 2, we consider problems of layout and weight constraints on a simple but representative model. Finally, in Section 3, we use this model to deal with fault-tolerance issues.

1 Energy savings by means of parallelism

Although we are using parallel processing, we are not interested here in reducing the time required by a task, but in reducing power consumption. If the CPU is considered as a capacitor-based system, then the power consumption is governed by the following physical law:

$$\frac{\text{energy}}{\text{time}} \propto \text{voltage}^2 \cdot \text{clock\_speed}.$$

Furthermore, a given task may be defined by its number of required CPU clock cycles, that is the elapsed time multiplied by the processor system clock speed, yielding:

$$\frac{\text{energy}}{\text{task}} \propto \text{voltage}^2.$$

If we consider, as in [5, 17], that the voltage is proportional to the speed (which is a valid first-order approximation), the parallel execution on P identical devices of a task allows to reduce the processor system speed by a factor of P, and therefore the energy spent on the task by P^2. In Figure 1, we

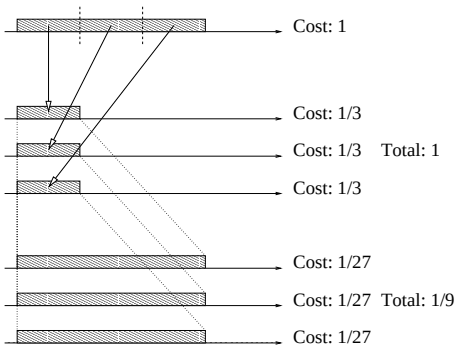

Figure 1: Use of parallelism to save energy in a simple configuration.

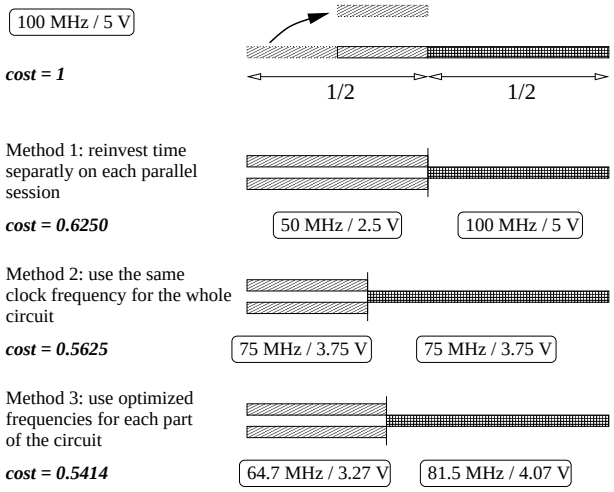

Figure 2: Different ways to exploit diversity of degrees of parallelism.

illustrate how a task parallelizable on three processors can be perform with a significantly lower consumption. In the upper part of the picture, we show the initial task with energy cost 1. In the middle part of it, the task is performed on three distinct processors. The cost remains the same since the total amount of work and the CPU clock frequency do not change. But on this example, the task is completed within a third of the original duration. Finally, in the lower part, we turn the gain in time into a gain in energy: the CPU clock speed and the voltage are decreased by a factor of three, so that the energy cost of the computation is divided by nine.

1.1 Restricted parallelism

It is clear that for many systems, not all the computation can be parallelized in an arbitrary number of processors. In Figure 2, we illustrate different strategies of turning the gain in time into a gain in power cost while having different degrees of parallelism.

The first method , called SEPARATE, assigns shares of time for the different parts of the calculus in the same manner as the sequential circuit.

In the example, the first part only was parallelizable. Since it had 50% of the time at the beginning, this proportion is kept, and the new design saves 75% of the energy on the first half of the computation, that is 3/8.

The second method , denoted UNIFORM, finds a uniform clock speed for the entire new circuit. Since the new system requires only 75% of the original clocks, the clock speed is globally multiplied by 3/4, and the total savings are $1 - (3/4)^2 = 7/16$.

The last method , OPTIM, finds an optimized clock speed on each section of the circuit to minimize energy consumption. In the example, the savings are of 45.9%.

Clearly, OPTIM achieves the better savings at the higher complexity level. However, it sounds quite difficult to handle such various values of clock frequency and voltage on the same chip. But how do SEPARATE and UNIFORM compare?

Result 1 *For any computation with different degrees of parallelism, the savings of UNIFORM are greater than the savings of SEPARATE.*

PROOF. Consider a circuit divided in two parts. The two parts respectively take a sequential time of t_1 and t_2 and are parallelizable in p_1 and p_2 processors. If the initial cost of the computation is $\mathcal{C} = k(t_1 + t_2)$, the energy cost of the SEPARATE strategy is then

$$\mathcal{C}_{separate} = k \cdot \left(\frac{t_1}{p_1^2} + \frac{t_2}{p_2^2} \right)$$

and the UNIFORM one gives

$$\mathcal{C}_{uniform} = k(t_1 + t_2) \left(\frac{t_1/p_1 + t_2/p_2}{t_1 + t_2} \right)^2 = \frac{k}{t_1 + t_2} \left(\frac{t_1}{p_1} + \frac{t_2}{p_2} \right)^2 .$$

And we have

$$\mathcal{C}_{separate} - \mathcal{C}_{uniform} = \frac{t_1 t_2 k (p_1 - p_2)^2}{p_1^2 p_2^2 (t_1 + t_2)} \geq 0$$

which achieves the proof for a two-part circuit.

We can then proceed as follows, we merge parts of the circuits two by two (i.e. give them the same clock frequency), until only one remains. Indeed, once two parts receive the same frequency as before, the block behaves as a part of sequential time of computation $t_1 + t_2$ and degree q of parallelism, where $\frac{t_1+t_2}{q} = \frac{t_1}{p_1} + \frac{t_2}{p_2}$. The presence of fractional "degrees" of parallelism does not affect the first part of the proof. □

As a result, the presence of a unique clock frequency simplifies the system while providing good savings in energy. Further studies have shown that for high diversities of parallelism, an OPTIM approach was interesting to take full advantage of the method, more especially when the highly parallel part takes a significant share of the task. We believe, however, that the nest of the method will take place when the parallelism is efficient but restricted.

We give in the following a more practical insight of the expected gains. Let t be the number of CPU clocks required to perform a given task (or, equivalently, the sequential time of execution). Suppose that t can be decomposed into $t = t_1 + t_2$, and t_1 represents the part of t which is parallelizable onto p parallel circuits. Following Amdahl's law [1], the speedup obtained by such a performance enhancement is given by:

$$\begin{aligned} \text{Speedup} &= \frac{\text{Execution time without the enhancement}}{\text{Execution time using the enhancement when possible}} \\ &= \frac{t_1 + t_2}{\frac{t_1}{p} + t_2} \end{aligned}$$

Ideally, if $t_2 = 0$, we have Speedup $= 1/p$. In a UNIFORM strategy, the energy is then multiplied by a factor $F = 1/\text{Speedup}^2$, which means:

$$F = \left(1 - \frac{t_1}{t_1 + t_2} \cdot \frac{p-1}{p}\right)^2$$

This equation is the basis of the Figure 3, which shows the typical expected gains. For instance, if 60% of the task can be computed by 4 processors, 70% of the energy is saved. If 40% of the task can be split into 2 processors, the savings represent 36% of the energy.

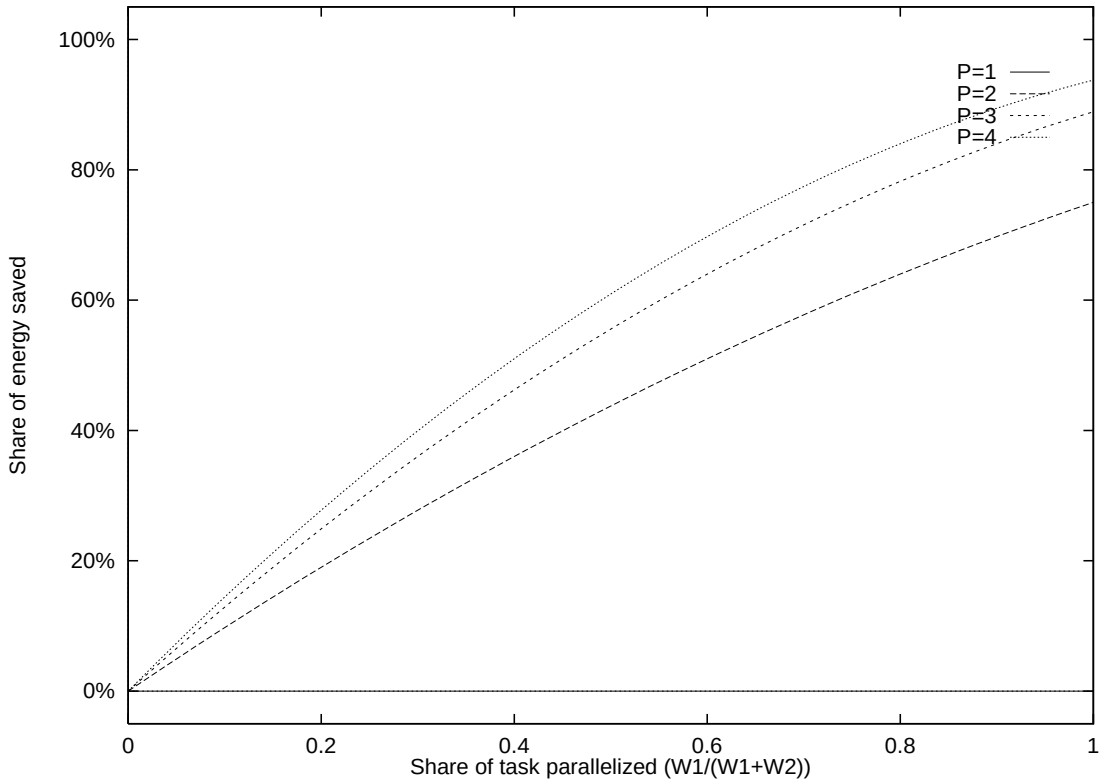

Figure 3: Energy savings for various numbers of processors.

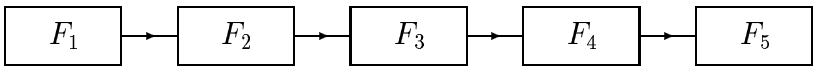

Figure 4: Modeling of a circuit.

2 Place constraints

On-board systems or mobile terminals have to satisfy huge constraints on the place devoted to the circuitry. We propose here a model along with two ways to optimize the systems. We consider a circuit which is a filter of serial elements $\mathcal{F} = (F_1, \ldots, F_p)$, as shown in Figure 4. This circuit is compared to a parallel version, shown in Figure 5. We consider here a pure "parallel" version of the circuit, that is, no interface is required to distribute the data. Of course this assumption is not verified in general, but we may add some elements in the filter to perform this operation. However, we have to take into consideration some limits on the parallelism of the circuit: an element F_i will be parallelized into at most p_i^{max} processors. We denote t_i the number of CPU clocks required to perform the stage i of the computation. Then, the amount of parallelism $(p_1, \ldots, p_n)$ assigned to the different parts of the filter can be determined by two types of optimization:

- layout constraints,
- minimization of the total weight of the system.

According to the above discussion, adopting a UNIFORM strategy, the total energy cost of the system is then proportional to the factor:

$$c = \left(\frac{\sum_{i=1}^{i=n} \frac{t_i}{p_i}}{\sum_{i=1}^{i=n} t_i} \right)^2 .$$

The two following sections address the two types of optimization.

2.1 Limited space

In this part, we consider that the whole circuitry has to be contained in a limited space (a chip for instance). Given this constraint, we aim at minimizing

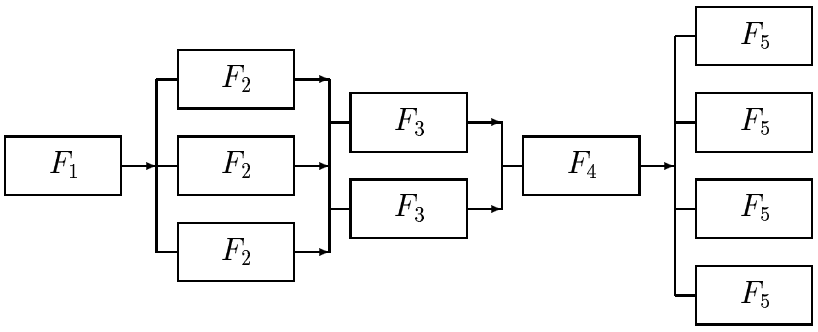

Figure 5: Parallel circuit.

the energy consumed by the filter. We also consider that a copy of circuit F_i takes an area c_i, while the total area available is c. The optimization problem can then be written as follows:

$$\text{Minimize } \sum_{i=1}^{i=n} \frac{t_i}{p_i} \text{ subject to } \begin{cases} p_i \leq p_i^{max}, & 1 \leq i \leq n \\ \sum_{i=1}^{i=n} c_i p_i \leq c \end{cases}$$

We have to keep in mind the fact that minimizing the total time of the computation is equivalent to maximize the energy savings, since a smaller number of CPU clocks can be turned into a cheaper computation in terms of energy taking an equal time. Furthermore, it is possible to convert this problem into a 0/1 knapsack problem. For instance, we can set

$$\begin{cases} A_{(i,j)} = \frac{t_i}{j} - \frac{t_i}{j+1} \\ K_{(i,j)} = c_i \end{cases}$$

for $1 \leq i \leq k$ and $1 \leq j \leq p_i^{max} - 1$. Then we can easily derive the solution of our problem when some $x_{(i,j)} \in \{0, 1\}$ are found, that solve:

$$\begin{cases} \text{Maximize} \sum A_{(i,j)} x_{(i,j)} \\ \text{Subject to} \sum K_{(i,j)} x_{(i,j)} \leq c \end{cases}$$

Indeed, since $K_{(i,j_1)} = K_{(i,j_2)}$ and $A_{(i,j_1)} > A_{(i,j_2)}$ as soon as $j_1 < j_2$, we have $x_{(i,j_2)} = 1 \Rightarrow x_{(i,j_1)} = 1$. We then set $p_i = \max\{j : x_{(i,j)} = 1\}$ which gives our result. Note that this derivation remains possible even when the speedups

are not necessarily linear. The only necessary property is $K_{(i,j_1)} \leq K_{(i,j_2)}$ and $A_{(i,j_1)} \geq A_{(i,j_2)}$ for $j_1 < j_2$.

Finding a set of values for the $x_{(i,j)}$'s such that $\sum A_{(i,j)}x_{(i,j)} \geq A$ for some A (and subject to the above constraint) is known as being NP-hard [10]. Since, reversely, a knapsack problem can be turned into our limited space problem (by setting as many circuits as objects, and $p_i^{max} := 2$), we are also addressing - in some way - a NP-hard problem (in fact, to be so, the number of clocks should grow in a non-polynomial fashion with respect to the size of the problem). However, until now, various methods have been proposed to solve efficiently this problem. They include dynamic programming [14, pp. 420-422], genetic algorithms [15], simulated annealing [7] and taboo search [6]. Note that dynamic programming is a pseudo-polynomial time algorithm for finding the optimal solution (i.e. the solution can be found in polynomial time if the number of clocks is polynomial with the size of the problem).

2.2 Minimal weight

We consider now an alternative problem. We suppose our system is on-board or mobile, and we aim at minimizing its weight. Indeed, several studies suggest that the cost of a satellite is proportional to its weight (essentially because the launch cost dominates) [9, 11]. We denote m_i the mass required for a single copy of the circuit F_i, and M_B the battery mass (or the solar panels' mass) assigned to the activity of the whole sequential version of circuit $\mathcal{F}$. We also assume that the power required is proportional to the battery mass (this assumption seems to be quite pessimistic; the power requires probably more mass while growing). Then our problem becomes:

$$\text{Minimize } \sum_{i=1}^{i=n} p_i m_i + M \cdot \left(\sum_{i=1}^{i=n} \frac{t_i}{p_i} \right)^2 \text{ subject to } p_i \leq p_i^{max}, \quad 1 \leq i \leq n$$

where $M := \frac{M_B}{\sum_{i=1}^{i=n} t_i}$.

This optimization criteria seems quite unnatural to solve, since it is far from being linear. The restricted problem with $m_i := 0$ leads directly to the above problem, with the same comments on NP-hardness. Nevertheless, the

function to minimize keeps good convexity properties, so that we can describe our problem in semi-definite linear programming. This domain is a natural extension of linear programming that allows the introduction of some more complex inequalities (see [13, 2]).

We add to the series $(p_i)_{1\leq i\leq n}$ a set of variables $(y_i)_{1\leq i\leq n}$, and t. We construct the matrix X as follows:

$$X = \begin{bmatrix} \begin{pmatrix} y_1 & \sqrt{t_1} \\ \sqrt{t_1} & p_1 \end{pmatrix} & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \\ 0 & \dots & \begin{pmatrix} y_n & \sqrt{t_n} \\ \sqrt{t_n} & p_n \end{pmatrix} & \\ 0 & & & \begin{pmatrix} 1 & \sum_{i=n}^{i=1} y_i \\ \sum_{i=n}^{i=1} y_i & t \end{pmatrix} \end{bmatrix}$$

The optimization problem is then to minimize $\sum_{i=n}^{i=1} p_i m_i + Mt$ under the conditions (1) $X \geq 0$ (i.e. X semi-definite positive) and (2) $(p_i)_{1\leq i\leq n}$ are integers. The first condition can be fixed by the following result:

Result 2 *There exist a polynomial-time algorithm that can find $(p_i)_{1\leq i\leq n}$ that minimize $\sum_{i=n}^{i=1} p_i m_i + M \cdot \left(\sum_{i=1}^{i=n} \frac{t_i}{p_i}\right)^2$ under the condition that each p_i is real-valued and greater than 1, for $1 \leq i \leq n$.*

PROOF. Obviously, the condition (1) $X \geq 0$ is equivalent to the set of inequalities:

$$\left| \begin{array}{ll} y_i \geq 0 & 1 \leq i \leq n \\ y_i \geq \frac{t_i}{p_i} & 1 \leq i \leq n \\ t \geq \left(\sum_{i=n}^{i=1} y_i\right)^2 & \end{array} \right.$$

This implies

$$t \geq \left(\sum_{i=n}^{i=1} \frac{t_i}{p_i}\right)^2$$

hence the reduction. Minimizing $\sum_{i=n}^{i=1} p_i m_i + Mt$ under the condition $X \geq 0$ is identified as a polynomial-time problem under reasonable constraints for the floating-point precision [4, on the EVP problem]. □

We still have to insure that the $(p_i)_{1 \leq i \leq n}$'s are integers. We can introduce a set of 0-1 variables $(c_i^j)_{1 \leq i \leq n, j \geq 0}$ and define $p_i = \sum_{j \geq 0} 2^j c_i^j$, which does not increase too much the amount of variables since the integers in question are small.

However, there exists a simple randomized linear-time algorithm that can address the problem without too many losses, as stated here.

Result 3 *There exists a linear-time randomized algorithm that can assign integer valued $\hat{p_i}$'s that verify*

$$E\left[\sum_{i=n}^{i=1} \hat{p_i} m_i + M \cdot \left(\sum_{i=1}^{i=n} \frac{t_i}{\hat{p_i}}\right)^2\right] \leq \frac{9}{8}\left[\sum_{i=n}^{i=1} p_i m_i + M \cdot \left(\sum_{i=1}^{i=n} \frac{t_i}{p_i}\right)^2\right]$$

where the p_i's are the solutions of the real-valued program previously defined.

PROOF. Once the values of the p_i's are obtained, we choose independently each $\hat{p_i}$ as a random variable according to the value of p_i. We note that we have in this case:

$$E\left[\sum_{i=n}^{i=1} \hat{p_i} m_i + M \cdot \left(\sum_{i=1}^{i=n} \frac{t_i}{\hat{p_i}}\right)^2\right] = \left| \begin{array}{l} \sum_{i=n}^{i=1} m_i E[\hat{p_i}] \\ +M \sum_{i=1}^{i=n} t_i E\left[\frac{1}{\hat{p_i}^2}\right] \\ +M \sum_{i=1}^{i=n} \sum_{j=i+1}^{i=n} t_i t_j E\left[\frac{1}{\hat{p_i}}\right] E\left[\frac{1}{\hat{p_j}}\right] \end{array} \right.$$

We set each $\hat{p_i}$ according to the following law:

$$\begin{array}{rcl} P[\hat{pi} = \lfloor p_i \rfloor] & = & \alpha \\ P[\hat{pi} = \lfloor p_i \rfloor + 1] & = & 1 - \alpha \end{array} \quad \text{where } \alpha = \frac{\lfloor p_i \rfloor (\lfloor p_i \rfloor + 1)}{p_i} - \lfloor p_i \rfloor$$

Then we can check that

$$\begin{aligned} E[\hat{p_i}] - p_i &= \alpha\lfloor p_i \rfloor + (1-\alpha)(\lfloor p_i \rfloor + 1) - p_i = p_i \frac{\alpha(1-\alpha)}{\lfloor p_i \rfloor (\lfloor p_1 \rfloor + 1)} \leq \frac{1}{8} p_i \\ &\quad \text{since } \lfloor p_i \rfloor \geq 1 \text{ and, for } 0 \leq \alpha \leq 1 \text{ we have } \alpha(1-\alpha) \leq \tfrac{1}{4}, \\ E[\frac{1}{\hat{p_i}}] - \frac{1}{p_i} &= \frac{\alpha}{\lfloor p_i \rfloor} + \frac{1-\alpha}{\lfloor p_i \rfloor + 1} - \frac{1}{p_i} = 0 \\ E[\frac{1}{\hat{p_i}^2}] - \frac{1}{p_i^2} &= \frac{\alpha}{\lfloor p_i \rfloor^2} + \frac{1-\alpha}{(\lfloor p_i \rfloor + 1)^2} - \frac{1}{p_i^2} = \frac{1}{p_i^2} \frac{\alpha(1-\alpha)}{(\lfloor p_i \rfloor + \alpha)^2} \leq \frac{1}{8} \frac{1}{p_i^2} \\ &\quad \text{for } \lfloor p_i \rfloor = 1, \text{ we observe that } \tfrac{\alpha(1-\alpha)}{(\alpha+1)^2} \leq \tfrac{1}{8} \text{ when } 0 \leq \alpha \leq 1. \\ &\quad \text{for } \lfloor p_i \rfloor \geq 2, \text{ we have } \alpha(1-\alpha) \leq \tfrac{1}{4}, \text{ as already noticed.} \end{aligned}$$

This achieves the proof, since the deviation term by term is never greater than $\frac{1}{8}$. □

In fact, an enhanced version of this proof allows to get the optimization factor to less than 1.12175 instead of 9/8.

In the past, various other methods have been introduced to solve this class of problems. They include branch&bound, genetic, taboo, and other heuristics. For small enough problems, an exhaustive search may be sufficient. For larger problems, the modeling itself may need further refinement, due to the limited degree of parallelism in practice.

3 Fault tolerance

One of the main concerns in satellite design is the fault tolerance of the system. It is strongly related to the life-time of a satellite. However, a common way to solve nowadays reliability problems consists in doubling every light-weight component on-board. Typically, the processors and other electronic devices will be doubled. We argue that parallelism can be introduced while taking care of this reliability concerns.

When one component in the satellite is doubled, there is a need for switching between the two components so that the non-faulty one is used. This technique has already been extended to devices that can select, for instance p

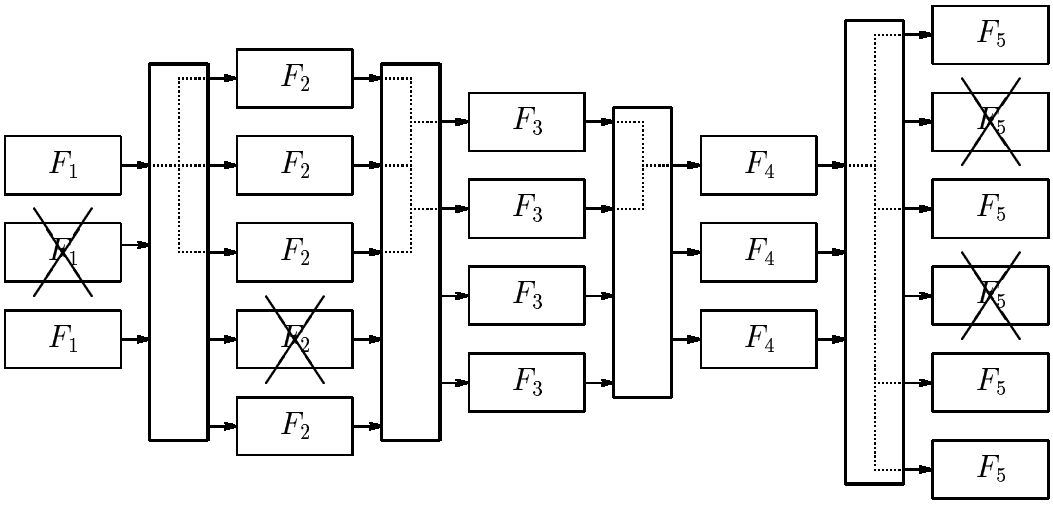

Figure 6: Fault-tolerant system using switching devices.

components out of $p+k$ possible, in order to tolerate k faults [3]. If the reliability of one copy of the part i of the filter is ρ_i (i.e. the probability it doesn't fail during the life-time of the satellite), then the reliability of the system becomes

$$\rho_i^{(k)} = \sum_{j=p_i}^{j=p_i+k} \left(\begin{array}{c} p_i + k \\ j \end{array} \right) \rho_i^j (1-\rho_i)^{p_i+k-j}$$

As a result, by increasing k, it is possible to improve the reliability up to a given factor. As an example, if we consider an approximatively equivalent amount of circuitry, we can compare the use of four independent copies of the sequential circuit (Figure 4), to two independent copies of the original parallel circuit (Figure 5), and to one copy of the fault-tolerant parallel version (Figure 6). Using an independent reliability of 95% for each component, and considering the switches of Figure 6 as fault-free, we obtain a global reliability of 99.7%, 81.4% and 99.6% respectively. Therefore, provided that good switches can be built, it is possible to achieve reasonable fault-tolerance even when using more circuitry.

4 Conclusion

The recent uprising of sophisticated mechanisms in mobile telecommunication systems has revealed the need for power-saving oriented devices. For instance, fax services on handy phones or on-board ATM switches for telecommunication satellites have been planed or implemented. We believe also that the environmental concerns will promote devices that consume less energy.

Surprisingly, the gains in energy due to the parallelism grow as square of the speedup, which opens a large field for future research. Directions of further work include:

- the parallelization of algorithms for on-board systems,
- the analysis of the efficiency of these approaches,
- the development of heuristics for the dimensioning of the degree of parallelism, inspired from Section 3,
- the enhancement of the reliability of these systems.

References

[1] G.M. Amdahl. Validity of the single-processor approach to achieving large scale computing capabilities. In *AFIPS Conference Proceedings*, volume 30, pages 483–485, Reston, Va., 1967. AFIPS Press.

[2] Egon Balas, Sebastian Ceria, and Gerard Cornuejols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, 58:295–324, 1993.

[3] Bruno Beauquier, Jean-Claude Bermond, Eric Darrot, Stéphane Perennes, and David Toth. On flow networks with output fault for satellite telecommunications. Personal communication, March 1998.

[4] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishman. *Linear Matrix Inequalities in System and Control Theory*. SIAM Studies in Applied Mathematics, Philadelphia, 1994.

[5] T. D. Burd and R. W. Brodersen. Energy efficient CMOS microprocessor design. In *Proc. 28th Hawaii Int'l Conf, on System Sciences*, volume 1, pages 288–297, January 1995.

[6] F. Dammeyer and S. Voss. Dynamic tabu list management using the reverse elimination method. *Annals of Operations Research*, 41:31–46, 1993.

[7] A. Drexl. A simulated annealing approach to the multiconstraint zero-one knapsack problem. *Computing*, 40:1–8, 1988.

[8] K Govil, E. Chan, and H. Wasserman. Comparing algorithms for dynamic speed-setting of a low-power cpu. In *Proceedings of the First Annual International Conference on Mobile Computing and Networking (MOBICOM'95)*, pages 13–25, November 1995.

[9] Cyrus D. Jilla and David W. Miller. Satellite design: past, present and future. *International journal of small satellite engineering*, december 1995. A good introduction to the small satellite technology.

[10] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of computer computations*, pages 85–103. Plenum-Press, New-York, 1972.

[11] W. J. Larson and J. R. Wertz, editors. *Space mission analysis and design.* Space technology series. Microcosm, Inc. and Kluwer Academic Publishers, second edition, 1992.

[12] J. R. Lorch and A. J. Smith. Reducing processor power consumption by improving processor time management in a single-user operating system. In *Proceedings of the Second Annual International Conference on Mobile Computing and Networking (MOBICOM'96)*, pages 143–154, November 1996.

[13] L. Lovasz and A. Schrijver. Cones of matrices and set-functions and 0-1optimization. *SIAM J. Optimization*, 1(2):166–190, May 1991.

[14] George L. Nemhauser and Laurence A. Wolsey. *Integer and combinatorial optimization.* Wiley, 1988.

[15] Jochen Thiel and Stephan Voss. Some experiences on solving multiconstraint zero-one knapsack problems with genetic algorithms. *Information Systems and Operational Research (INFOR)*, 32(4):226–242, November 1994.

[16] Gyanendra Tiwary. Reducing power consumption in ASIC's. *Computer Design*, March 1995.

[17] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced CPU energy. In *Proc. 1st USENIX Symp. on Operating Systems Design and Implementation*, pages 13–23, November 1994.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399